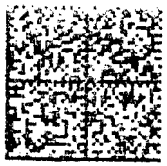
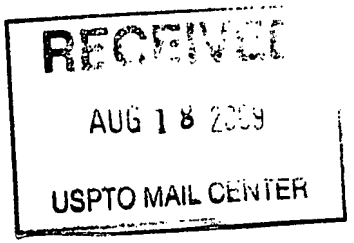
Organization_____ Bldg/Room_____
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
If Undeliverable Return in Ten Days

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300



UNITED STATES POSTAGE
$ 01.56⁰
AUG 10 2009
MAILED FROM ZIP CODE 22314

AN EQUAL OPPORTUNITY EMPLOYER



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/749,024	12/30/2003	Parimal Pal Chaudhuri	075005.0102	9229

7590 08/10/2009

SOMSHBHUBRO PAL CHOUDHURY
NETGEAR INC.
4500 GREAT AMERICAN PARKWAY
SANTA CLARA, CA 95054

EXAMINER	
YALEW, FIKREMARIAM A	

ART UNIT	PAPER NUMBER
2436	

MAIL DATE	DELIVERY MODE
08/10/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Advisory Action Before the Filing of an Appeal Brief	Application No. 10/749,024	Applicant(s) CHAUDHURI, PARIMAL PAL	
	Examiner Fikremariam Yalew	Art Unit 2436	

–The MAILING DATE of this communication appears on the cover sheet with the correspondence address –

THE REPLY FILED 09 June 2009 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE.

1. ☒ The reply was filed after a final rejection, but prior to or on the same day as filing a Notice of Appeal. To avoid abandonment of this application, applicant must timely file one of the following replies: (1) an amendment, affidavit, or other evidence, which places the application in condition for allowance; (2) a Notice of Appeal (with appeal fee) in compliance with 37 CFR 41.31; or (3) a Request for Continued Examination (RCE) in compliance with 37 CFR 1.114. The reply must be filed within one of the following time periods:
   a) ☒ The period for reply expires 3 months from the mailing date of the final rejection.
   b) ☒ The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of the final rejection.

   Examiner Note: If box 1 is checked, check either box (a) or (b). ONLY CHECK BOX (b) WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. Any reply received by the Office later than three months after the mailing date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

NOTICE OF APPEAL

2. ☐ The Notice of Appeal was filed on _____. A brief in compliance with 37 CFR 41.37 must be filed within two months of the date of filing the Notice of Appeal (37 CFR 41.37(a)), or any extension thereof (37 CFR 41.37(e)), to avoid dismissal of the appeal. Since a Notice of Appeal has been filed, any reply must be filed within the time period set forth in 37 CFR 41.37(a).

AMENDMENTS

3. ☐ The proposed amendment(s) filed after a final rejection, but prior to the date of filing a brief, will not be entered because
   (a) ☐ They raise new issues that would require further consideration and/or search (see NOTE below);
   (b) ☐ They raise the issue of new matter (see NOTE below);
   (c) ☐ They are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or
   (d) ☐ They present additional claims without canceling a corresponding number of finally rejected claims.

   NOTE: _____. (See 37 CFR 1.116 and 41.33(a)).
4. ☐ The amendments are not in compliance with 37 CFR 1.121. See attached Notice of Non-Compliant Amendment (PTOL-324).
5. ☐ Applicant's reply has overcome the following rejection(s): _____.
6. ☐ Newly proposed or amended claim(s) _____ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).
7. ☒ For purposes of appeal, the proposed amendment(s): a) ☒ will not be entered, or b) ☐ will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.
   The status of the claim(s) is (or will be) as follows:
   Claim(s) allowed: _____.
   Claim(s) objected to: _____.
   Claim(s) rejected: 1-30.
   Claim(s) withdrawn from consideration: _____.

AFFIDAVIT OR OTHER EVIDENCE

8. ☐ The affidavit or other evidence filed after a final action, but before or on the date of filing a Notice of Appeal will not be entered because applicant failed to provide a showing of good and sufficient reasons why the affidavit or other evidence is necessary and was not earlier presented. See 37 CFR 1.116(e).
9. ☐ The affidavit or other evidence filed after the date of filing a Notice of Appeal, but prior to the date of filing a brief, will not be entered because the affidavit or other evidence failed to overcome all rejections under appeal and/or appellant fails to provide a showing a good and sufficient reasons why it is necessary and was not earlier presented. See 37 CFR 41.33(d)(1).
10. ☐ The affidavit or other evidence is entered. An explanation of the status of the claims after entry is below or attached.

REQUEST FOR RECONSIDERATION/OTHER

11. ☒ The request for reconsideration has been considered but does NOT place the application in condition for allowance because: See Continuation Sheet.
12. ☐ Note the attached Information *Disclosure Statement*(s). (PTO/SB/08) Paper No(s). _____
13. ☐ Other: _____.

/Nasser G Moazzami/
Supervisory Patent Examiner, Art Unit 2436

Continuation of 11. does NOT place the application in condition for allowance because: The applicant not properley complaing with the Oath since there are 5 other inventors in this invention and the applicant fail to disclose all the others inventor information which required to by Oath.Therefore the examiner suggest that the applicant's has to submit all the others information to prosecute this case further.

Cellular Automata Based Cryptosystem (CAC)

Subhayan Sen[1], Chandrama Shaw[1], Dipanwita Roy Chowdhuri[2], Niloy Ganguly[3], and P. Pal Chaudhuri[1]

[1] Department of Computer Science & Technology, Bengal Engineering College (D U) Howrah, India 711103
{subhayan@,cshaw@,ppc@ppc.}becs.ac.in
[2] Department of Computer Science & Technology, Indian Institute of Technology Kharagpur, India 721302
drc@cse.iitkgp.ernet.in
[3] Computer centre, IISWBM, Calcutta, India 700073
n_ganguly@hotmail.com

Abstract. This paper introduces a *Cellular Automata* (CA) based symmetric key cryptosystem for block cipher. The scheme named as CAC (Cellular Automata based Cryptosystem) employs a series of transforms – simple, moderately complex, and complex – all generated with different classes of CA. CAC provides a low cost, high speed cryptosystem with desired level of security. Cryptanalysis of the proposed scheme is reported along with similar analysis for two popular systems - DES and AES. Experimental results confirm that the security of the system is significantly better than DES and comparable to that of AES. The encryption/decryption throughput is higher than that of both AES and DES.

1 Introduction

This paper reports a high speed, low cost cryptosystem with desired level of security. Its hardwired version supports real time encryption/decryption. The scheme referred to as CAC (Cellular Automata based Cryptosystem) employs different classes of transforms generated with Cellular Automata (CA).

We currently live in an internetworked society where a large volume of different classes of data travel around the globe. This electronic data transmission should be secured enough against unwanted interceptor. In the above context we aim to achieve the following objectives for design of CAC: (i) High speed operation, specifically on line real time data encryption/decryption; (ii) low cost of implementation; and (iii) acceptable level of security.

In this paper, we concentrate on developing an innovative cryptosystem based on the theory of *Cellular Automata*(CA). A CA can be viewed as a parallel machine simulating a discrete dynamical system. Further, the inherent parallelism of CA cells with their simplicity and local interactions make it particularly suitable for designing a low-cost crypto-hardware. The above mentioned advantages have lead researchers to design various Cellular Automata based cryptosystems

R. Deng et al. (Eds.): ICICS 2002, LNCS 2513, pp. 303–314, 2002.

[2–5]. In [2] Cellular Automata is used as random sequence generator. In [3], non-homogeneous Cellular Automata has been proposed for *public-key* cryptosystems. Gutowitz [4] uses Cellular Automata as discrete dynamical system to add complexity of the cryptosystem. But none of these schemes has been able to withstand the modern attacks developed out of the cryptanalysis techniques [1]. Cellular Automata based block cipher and stream cipher schemes are also presented in [5]. But the scheme is insecure because of its inability to change the key. The ability to change the key is essential for any cipher. Also the scheme, as pointed out in [6], generates a subgroup of *affine group* and not the *alternating group*. In [7] another CA based block cipher scheme was proposed. But this is also unable to come out from the affine group constraint and so fails to achieve the desired level of security. This paper removes this bottleneck while generating non-affine CA transform.

The CA based cryptosystem (CAC) along with the encryption and decryption algorithm is outlined in *Section 3* after introducing CA preliminaries in *Section 2*. Discussion on cryptanalysis of CAC are covered in *Section 4*. A comparative study with other symmetric key block-cipher like DES and AES has also been included in this section. Finally, a low cost pipelined architecture of CAC crypto-hardware is reported in *Section 5*.

2 Cellular Automata Preliminaries

2.1 Introduction to $GF(2)$ CA

A CA consists of a number of cells arranged in a regular manner, where the state transitions of each cell depends on the states of its neighbors. The next state of a particular cell, as shown in Figure 1, is assumed to depend only on itself and on its two neighbors (left and right) and this leads to 3-neighborhood dependency. The state $q \in \{0,1\}$ of the i^{th} cell at time $(t+1)$ is denoted as $q_i^{t+1} = f(q_{i-1}^t, q_i^t, q_{i+1}^t)$, where q_i^t denotes the state of the i^{th} cell at time t and f is the next state function called the rule of the automata [8]. Since f is a function of 3 variables, there are 2^{2^3} or 256 possible next state functions. The decimal equivalent of the output column in the truth table of the function, as noted below is denoted as the rule number [8].

Neighborhood :	111	110	101	100	011	010	001	000	*RuleNo*
(i) NextState :	0	1	0	1	1	0	1	0	90
(ii) NextState :	1	0	0	1	0	1	1	0	150

A CA employing both XOR and XNOR local rules for different cells are referred to as *Additive CA*, while the ones using only XOR rules are noted as *Linear CA*. This class of CA is referred to as $GF(2)$ CA in the sense that each of the CA cells can store an element 0 or 1 in $GF(2)$. comprehensive treatment of $GF(2)$ CA results is noted in the book [8].

We next generalize this structure to study of $GF(2^p)$ CA [10] where each cell is capable of processing a symbol of $\{0,1,\cdots 2^p-1\} \in \mathrm{GF}(2^p)$. CAC employs $GF(2^p)$ CA that can be analyzed with the theory of extension field [9].

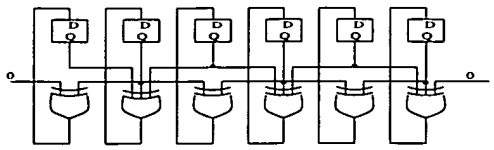


Fig. 1. A One Dimensional null boundary 6 cell Additive CA with rule vector <90,150,90,150,90,150>

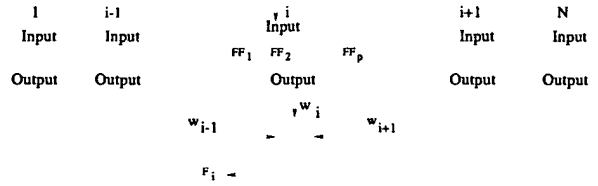


Fig. 2. General structure of a $GF(2^p)$ *Cellular Automata* machine

2.2 Introduction of $GF(2^p)$ CA

The *Fig.2* depicts the general structure of an n-cell $GF(2^p)$ CA. The connection among the cells follow a three neighborhood dependency in the sense that the next state $q_i(t+1)$ of the i^{th} cell depends upon the present states of $(i-1)^{th}$, i^{th} and $(i+1)^{th}$. The connection among the cells of the CA are weighted in the sense that to arrive at the next state $q_i(t+1)$ of i^{th} cell, the present states of $(i-1)^{th}$, i^{th} and $(i+1)^{th}$ are multiplied respectively with w_{i-1}, w_i and w_{i+1} and then added. In $GF(2^p)$ CA each cell, having p number of FFs(Flip-Flops), can store values 0,1,2,$\cdots$,(2^p-1) and the weights being elements of $GF(2^p)$.

If all the states in the state transition diagram of a CA lie in some cycles, it is a group CA; otherwise it is a non-group CA. Group CA can further be classified into *maximum* and *non-maximum* length CA. An n-cell *maximum*-length $GF(2^p)$ CA is characterized by the presence of a cycle of length (2^{pn}-1) with all nonzero states. On the other hand, a non-maximum length CA state transition diagram has a number of cycles.

3 Cellular Automata Based Cryptosystem (CAC)

The objectives of high speed of operation with lower implementation cost achieving high level of security are conflicting in nature. In order to meet such conflicting demands we apply a series of transforms of increasing complexity in successive levels. For the current vesion of CAC, four levels of transforms, as shown in *Fig. 3*, have been employed. The basic guiding factor is to achive a trade off in typical engineering design – realize the targeted objective with higher efficiency while minimizing cost. With a similar analogy, we apply low cost high speed linear and affine transforms in first two levels, while introducing complex non-affine transform at the third level to achive higher level of security. The fourth level is responsible for key-mixing.

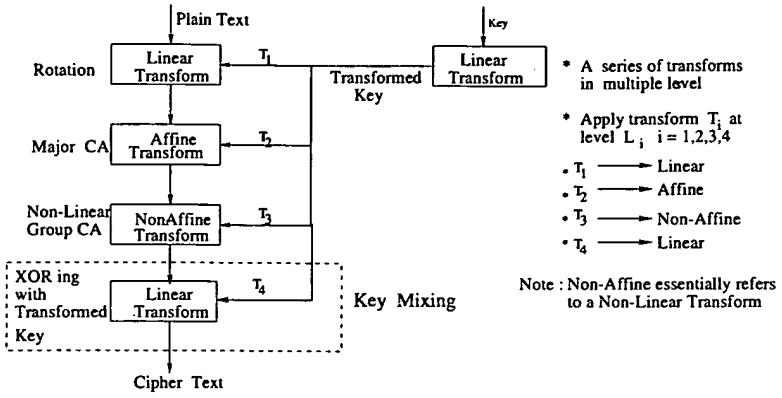


Fig. 3. Design of the encryption scheme

3.1 A Specific Implementation

A specific CAC implementation is shown in *Figure 4*. Four different levels of transformations are explicitly marked as *Level 1, 2, 3 and 4*. Different stages of computation are marked as (I), (II), (III), (IV), (V) and (VI) in *Figure 4*.

The encryption algorithm is based on two different classes of group CA, 16 cell $GF(2^8)$ *Major CA* and 16 cell $GF(2^8)$ *Minor CA*. The *Major CA* is a non-maximum length group CA with equal cycles of length 32. The *Minor CA* is a maximum-length CA. CAC with 16 cell $GF(2^8)$ CA can encrypt 16×8=128 bits of token at a time. Thus the token ($\mathcal{T}$) size and also the key size are taken as 128 bits.

Note: *The size of the key and token can be adjusted by changing the number of cells of the CA and/or the value of p in* $GF(2^p)$. *The basic scheme does not get effected by that.*

The operation of the encryption and decryption scheme is presented below. Each step of the operation is explained with the help of the *Figure 4*.

Level 1 – Linear Transformation on Key: The key (K) used for CAC scheme is a bit string of length same as the number of bits used for *Minor CA*. The input key is used as the initial seed of the *Minor CA*.

Role of *Minor CA*: The *Minor CA* is operated for a fixed number of clock cycles (d) for input of each token. Initially, the seed of the *Minor CA* (S_0) is the key K (marked as I in the *Fig 4*). For each successive input token, *Minor CA* generates a new state (marked as S_N) after running d number of steps from its current state (shown as II in *Figure 4*). The state S_N is utilized for four different purposes:

1. Provides the value δ by which each byte of the input token ($\mathcal{T}$) is rotated.
2. Provides seed for equivalent *Major CA* synthesis.
3. Provides the number of clock cycles (Δ) of *Major CA* operation for encryption.

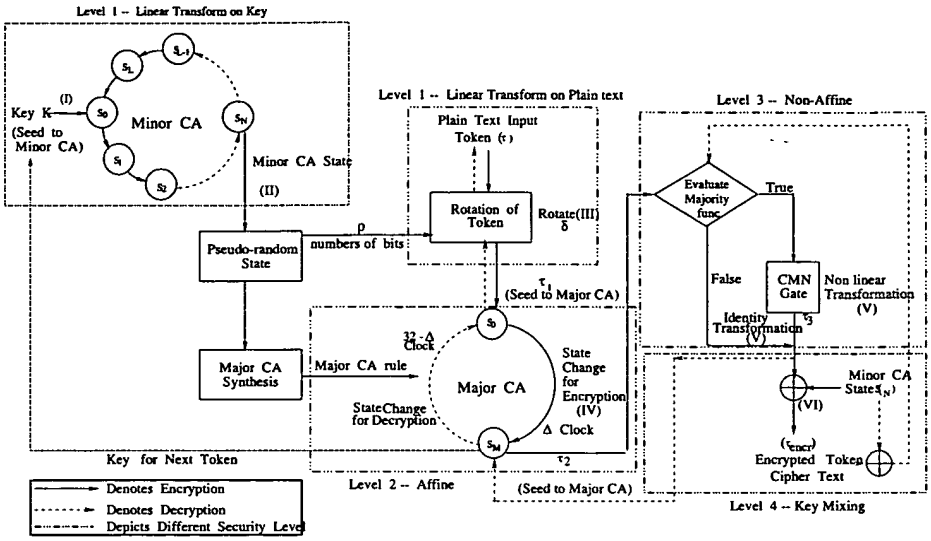


Fig. 4. Total encryption and decryption scheme

4. XORing the intermediate encrypted token to form the final encrypted token $\mathcal{T}_{encr}$.

Level 1 – Linear Transform on Token: The linear transformation of the token $\mathcal{T}$ to $\mathcal{T}_1$ is executed by rotating each byte of $\mathcal{T}$ by δ amount of steps (III in *Figure 4*). In decryption side token generated from $Major\ CA$ is subjected to a same amount of rotation in the opposite direction.

Level 2 – Affine Transform: Next we give an affine transform to the token $\mathcal{T}_1$ by using the $Major\ CA$. The $Major\ CA$ is generated at runtime by an efficient synthesis algorithm [10]. The $Major\ CA$ uses the input token ($\mathcal{T}_1$) as its seed and operates for Δ number of cycles to generate the encrypted token $\mathcal{T}_2$ (IV in *Figure 4*). The $Major\ CA$ has cycles of equal length 32. So, the $Major\ CA$ will invariably return back the input token $\mathcal{T}_1$ after running for 32 number of clock cycles. So the original token is returned after running the $Major\ CA$ for (32-Δ) clock cycles at the decryption side.

Level 3 – Non-Affine Transform: A non-affine transform is achieved by selective use of *Control Majority Not* (CMN) gate. CMN gate is a non-linear reversible gate with four inputs(1 data input and 3 control inputs) and one output. We will denote the control bits by c_1, c_2 and c_3. The function is defined as

$$y = x \oplus \{(c_1 \cdot c_2) \oplus (c_2 \cdot c_3) \oplus (c_3 \cdot c_1)\}$$

where $\oplus$ denote the XOR and $\cdot$ denote the AND function. The token $\mathcal{T}_2$ is subjected to CMN gate depending on the result of a function called *Majority Evaluation Function*. The Majority Evaluation Function takes the 5 bits, referred to as fixed-bits, of $\mathcal{T}_2$ and calculate the number of 1's in these bits. The 5 bit

positions are selected depending on S_N(*Figure 4*). If the number of 1's is greater than 2 then each bit of $\mathcal{T}_2$ except these fixed-bits are subjected to CMN gate. Otherwise, $\mathcal{T}_2$ remains as it is. In any case, we call the resultant token as $\mathcal{T}_3$ (V in *Figure 4*). Two sets of control bits taken from S_N applied to the CMN gate alternately. The fixed-bits have to be remained fixed because during decryption the same fixed-bits will require to get the same result from majority evaluation function.

Level 4 – Key Mixing: To enhance the security and randomness, we generate final encrypted token $\mathcal{T}_{encr}$ by XORing the *Minor CA* state S_N with the token $\mathcal{T}_3$(VI in *Figure 4*).

The algorithm for encryption and decryption process is presented next.

Algorithm 1 Encryption

Input: input file to be encrypted
K=key
Output: encrypted file
begin
Step 1. *Divide the file into 128 bit tokens ($\mathcal{T}$).*
Step 2. *Load initial seed of Minor CA S_0=K*
For each token $\mathcal{T}$ begin loop
Step 3. *Run the Minor CA for d time steps and obtain S_N*
Step 4. *Obtain δ from S_N. Rotate $\mathcal{T}$ by δ number of times and obtain $\mathcal{T}_1$*
Step 5a. *Randomly synthesize Major CA (CA_{maj}) using S_N as seed*
Step 5b. *Obtain Δ from S_N*
Step 5c. *Run CA_{maj} for Δ time steps with $\mathcal{T}_1$ as seed to obtain $\mathcal{T}_2$*
Step 5d. *S_0=$\mathcal{T}_2$*
if $\mathcal{T}_2$ satisfies MEF
Step 6a. *Obtain the 2 sets of Control bits for CMN gate*
Step 6b. *Apply CMN gate to non-fixed bits of $\mathcal{T}_2$ using the Control bits alternately*
Step 6c. *Assign the result to $\mathcal{T}_3$*
end if
Step 7. *XOR $\mathcal{T}_3$ with S_N to get $\mathcal{T}_{encr}$*
Step 8. *write $\mathcal{T}_{encr}$ in output file*
Go to Step 3 *untill the input file is exhausted*
end

Algorithm 2 Decryption

Input: input file to be decrypted
K=key
Output: decrypted file
begin
Step 1. *Divide the file into 128 bit tokens ($\mathcal{T}_{encr}$)*
Step 2. *Load initial seed of Minor CA S_0=K*
For each token $\mathcal{T}_{encr}$ begin loop
Step 3. *Run the Minor CA for d time steps and obtain S_N*
Step 4. *XOR $\mathcal{T}_{encr}$ with S_N to get $\mathcal{T}_3$*
if $\mathcal{T}_3$ satisfies MEF

Step 5a. *Obtain the 2 sets of Control bits for CMN gate*
Step 5b. *Apply CMN gate to non-fixed bits of $\mathcal{T}_3$ using the Control bits alternately*
Step 5c. *Assign the result to $\mathcal{T}_2$*
end if
Step 6a. $S_0 = \mathcal{T}_2$
Step 6b. *Randomly synthesize Major CA (CA_{maj}) using S_N as seed*
Step 6c. *Obtain Δ from S_N*
Step 6d. *Run CA_{maj} for 32-Δ time steps with $\mathcal{T}_2$ as seed to obtain $\mathcal{T}_1$*
Step 7. *Obtain δ from S_N. Rotate $\mathcal{T}_1$ by δ number of times in reverse order and obtain $\mathcal{T}$.*
Step 8. *Write back $\mathcal{T}$ into output file*
Go to Step 3 *untill the input file is exhausted*
end

4 Analysis of CAC

4.1 Different Levels of Security

Large Key Space: The number of possible key is very large (2^{128}) and all key are equiprobable to occur. This randomness in key generation gives random probability distribution in key space. Since we can change the size of the minor and major CA the key size can also vary. So we can have a variable key space of any arbitrary size.

Security Level 1 – Linear Transformation: Each byte of token $\mathcal{T}$ is subjected to a random rotation decided by *Minor CA* state. Since *Minor CA* is an excellent pseudo-random generator [8], this rotation of token introduces a degree of randomness to the input token.

Security Level 2 – Affine Transformation and On-line Synthesis of Major CA: The state transition of a *Major CA* which is additive generates an *affine transformation.*

On the fly generation of *Major CA* reduces the memory requirement by a large amount and as well as enhances the security. The number of all possible CA having the cycle structure of *Major CA* is higher than 2^{128} [10]. Thus, each seed (S_N) produces different *Major CA* providing us with the huge possibility of 2^{128} different *Major CA*. This ensures that each key value (K) will encrypt differently and no key will be superfluous. Thus the CAC satisfies one of the important criterion of a secure cryptosystem. The criterion is specified by the following theorem:

Theorem: [11] A necessary condition for a cryptosystem to have a perfect secrecy is that it to have at least as many keys as messages.

Security Level 3 – Non-affine Transformation: This is a non-affine reversible CA transform which enables CAC to generate a non-affine group which is the *alternating group.* The affine group is a small subgroup of the alternating group(*Fig. 5*). The analytical proof that the CAC scheme generates alternating

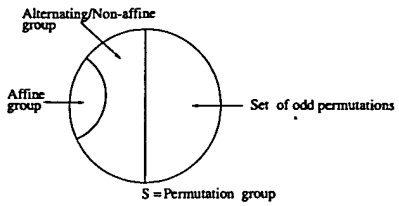


Fig. 5. Description of the permutation group

group is quite exhaustive and so omitted for short of space. Thus CAC, being able to generate the alternating group which is much larger than the affine group, satisfies another important criterion of a secure cryptosystem which says that ability to generate the alternating group on the message space is one of the strongest security conditions.

Security Level 4 – Key-mixing: The intermediate token ($\mathcal{T}_3$) is next XORed with the state S_N of the minor CA. This is very simple and takes only a single clock cycle. But it makes the encrypted token ($\mathcal{T}_{encr}$) totally unpredictable. The only way to return back to the original string is to randomly try with 128 bits which will cost $O(2^{128})$ operations, for every token.

Security Level 5: In order to further increase the level of security, our scheme can be used in bricklaying mode which will use multilevel encryption. This can be done with a minor increment of the cost while using the same basic structure reported in *Fig. 4*.

4.2 Cryptanalysis of CAC

The acceptance of any cryptosystem depends on its sustainability against various cryptanalysis attacks. Most important cryptanalysis are differential cryptanalysis [1] and Shannon's notion of perfect secrecy test [12]. We perform both these tests on CAC and as well as on DES and AES for the sake of comparison.

Results of Differential Cryptanalysis: We perform differential cryptanalysis with 50 different files having 11 different size. For each file, we take different fixed input differences to get the output probability distributions and the average value of the standard deviations for them is calculated. We also perform the same for DES and AES systems. The results are reported in *Table 1*. *Column II* of *Table 1* depicts the average mean standard deviation for CAC, where the same for DES and AES noted in *Column III* and *Column IV* respectively. The results for the current version of CAC is significantly better than that of DES and comparable to AES. It can also be noted from the results that the percentage of the standard deviation is around 4.0 whereas 10% is sufficient for a system to be considered as secured.

Results for Shannon's Security Quotient: We perform the Shannon's security test on CAC with 50 files for 9 different size and also perform the same on other cryptosystems (DES, AES) for the sake of comparison. *Column II* of

Table 1. Differential Cryptanalysis of our scheme and Comparative Study with DES and AES

Input file size (MB)	*Avg. Std. Devi*^{n} of XOR distributions for CAC (%)	*Avg. Std. Devi*^{n} of XOR distributions for DES (%)	*Avg. Std. Devi*^{n} of XOR distributions for AES (%)
1	4.36	31.95	4.2
2	4.30	30.03	4.0
4	4.26	29.05	3.63
6	4.17	28.24	3.62
8	3.91	28.10	3.67
10	4.02	28.89	3.52
12	3.89	28.74	3.51
14	3.55	28.52	3.48
16	3.40	27.86	3.43
18	3.42	27.74	3.26
20	3.59	27.67	3.24

Table 2. Measurement of Shannon's Security Quotient and comparative study with DES and AES

Input file size (MB)	Shannon's Security Quotient (ϑ) of CAC(%)	Shannon's Security Quotient (ϑ) for DES(%)	Shannon's Security Quotient (ϑ) for AES(%)
2	14.1605	14.2374	14.2345
3	11.5527	11.5531	11.5706
4	10.1060	10.2507	10.1675
7	7.5640	7.9141	7.6014
8	7.1182	7.1468	7.7046
9	6.7043	6.7139	6.7136
13	5.5868	5.5645	6.0266
14	5.3636	5.4001	5.4625
15	5.2097	5.3157	5.5552

Table 2 gives the average value of Security Quotient for our scheme calculated for different keys on each file size. The results show that our scheme fulfills the primary security level defined for any secure cryptosystem. *Column III and IV* of *Table 2* report the Security Quotient for DES and AES respectively, which establishes that CAC is better than DES and AES as far as Shannon's security notion is concerned.

4.3 Execution Time of Software Version

The main attractive feature of our CA based encryption scheme is its high speed of operation. Cellular automata are inherently parallel, so higher speed of execution of CAC is a natural outcome.

We have developed non-optimized reference code for CAC. Both the CAC and AES are run under same environment of $P-III$, $633MHz$ processor to

Table 3. Comparison of time of software version of CAC and AES

Input file size (in MB)	CAC Reference Code(in Sec)	AES Reference Code(in Sec)	AES Optimized Code(in Sec)
1.00	2.70	10.00	0.87
2.00	5.00	25.20	0.89
3.00	7.00	36.40	1.90
4.24	9.80	42.36	2.25
5.14	11.00	56.78	2.79
6.108	11.30	59.34	3.2
7.125	16.00	79.86	3.4
8.00	17.91	87.10	3.9
9.76	23.30	116.67	4.0
10.30	23.7	121.53	5.11
11.40	27.40	136.40	5.20
12.00	27.90	140.21	5.4

generate the results of *Table 3*. CAC reference code can be found to be significantly faster than that of AES. The optimized CAC code for commercial application is being developed. Preliminary results indicate that the optimized code of CAC will be faster than that of AES. However, the main advantage of CAC can be derived from its hardware version which is presented in the next section.

5 Cryto-Hardware Based On CAC

The pipelined architecture of CAC hardware is shown in *Figure 6*. The data path has five stages as explained below:

Stage 1 – Minor CA Implementation: It accepts the input key and control signals from *Control Block* and *CA Synthesis Hardware* block for on the fly generation of minor CA.

Stage 2 – Barrel Shifter Implementation: Its input register accepts the plain text token. The shift control of Barrel Shifter comes from *Stage 1*.

Stage 3 – Major CA Implementation with 3 Sub-Blocks: The three sub-blocks of this stage are flip-flops, a set of switches to implement Programmable $CA(PCA)$ and an array of XOR gates. The control of the PCA to generate different major CA comes from the CA synthesis hardware.

Stage 4: It covers the implementation of CMN(Control Majority Not) logic along with evaluation of majority function on the pseudo exhaustive fields of major CA.

Stage 5 – The XOR operation: The input to this stage is the token coming from Stage 3 and the minor CA state.

Two inter-stage pipeline registers are introduced between Stages 3 and 4, and also between 4 and 5. Different features of CAC crypto-hardware are next reported:

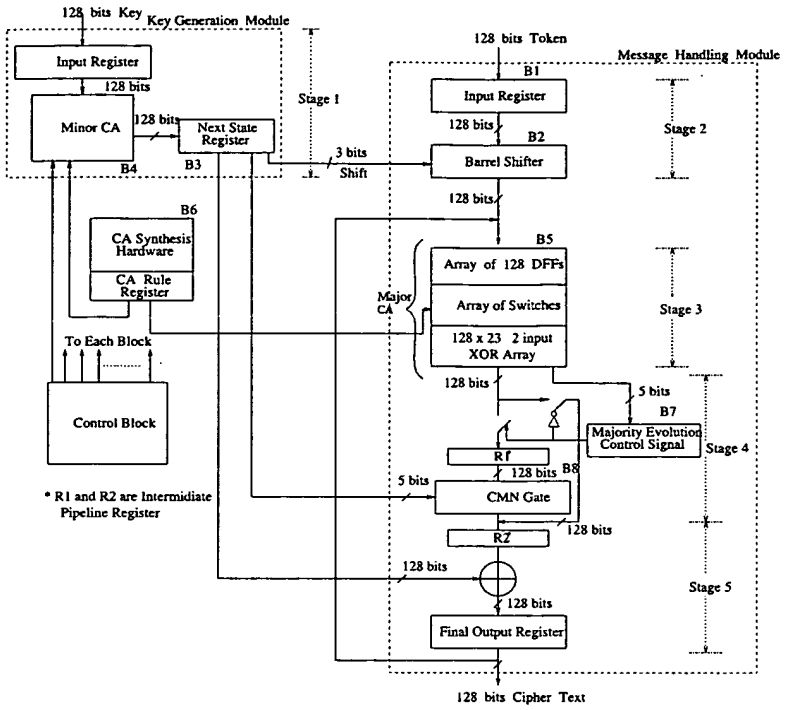


Fig. 6. Block Diagram *CAC* Hardware

- A verilog code has been written for the design and simulated using Cadence Verilog Simulator on a Sun Ultra-60 machine.
- The design has been synthesized and analyzed using Synopsis Design Compiler and Signal Scan.
- The design has been implemented with $0.25\mu CMOS$ technology.
- The pre-layout area estimate of the non-optimized design is 4.25×10^6 sq. micron.
- Static timing analysis of one complete run of *CAC* implementation on 128 bit plain text confirms correct operation of each stage with 1 GHz clock.
- For multiple rounds of operation(for the current implementation it is 4), the pipe line stages gets extended 4 times.
- The pipelined crypto-hardware throughput as per above timing analysis is 128 Gb/sec.
  Note: (i) Even if we assume 25% reduction of throughput for delay associated with silicon implementation, the throughput will be close to 100 Gb/sec. (ii) By contrast the full round Rijndael chip produced by NSA(National Security Agency) on $0.5CMOS$ *technology exhibit throughput of 5.7 Gb/sec. This is much lesser than the throughput of full round CAC chip.*
- Key generation hardware has been integrated within *CAC* implementation.

6 Conclusion

The CA based cryptosystem presented in this paper shows a very low cost, high speed encryption scheme with very high cracking complexity. The different cryptanalytic tests on our scheme shows that it satisfies primary security criterion and better than DES, AES. Its throughput is better than that of AES. The hardware version of CAC suits ideally for real time applications.

References

1. E. Biham and A. Shamir, "Differential cryptanalysis of des-like cryptosystems," *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.
2. S. Wolfram, "Cryptography with cellular automata," *Proceedings of Crypto*, pp. 429–432, 1985.
3. P. Guan, "Cellular automaton public-key cryptosystem," *Complex System*, vol. 1, pp. 51–57, 1987.
4. H. Gutowitz, "Cryptography with dynamical systems," *ESPCI*, 1995.
5. S. Nandi, B.K. Kar, P.P. Chaudhuri "Theory and Application of Cellular Automata in Cryptography," *IEEE Trans. Computers, Vol. 43*, Dec. 1994.
6. S. R. Blackburn, S. Murphy, K. G. P. I. S. Group, and R. Holloway, "Comments on 'theory and applications of cellular automata in cryptography'," *IEEE Transaction on Computers*, 1999.
7. N. Ganguly, A. Das, B. K. Sikdar, and P. P. Chaudhuri, "Cellular automata model for cryptosystem," *Cellular Automata Conference, Yokohama National University, Japan*, 2000.
8. P. P. Chaudhuri and et. al., "Additive cellular automata theory and applications, vol 1" *IEEE Computer Society Press, California, USA*, 1997.
9. T. R. N. Rao and E. Fujiwara, "Error-control coding for computer systems," *Prentice-Hall, Englewood Cliffs, N.J.*
10. N. Ganguly, "Cellular automata evolution - theory and applications," *PhD Thesis (proposed), Bengal Engineering College, (D U), Howrah, India*, 2002.
11. D. Welsh, "Codes and cryptography," *Clarendon Press, Oxford*, 1988.
12. C. E. Shannon, "Communication theory of secrecy systems," *Bell Systems Technical Journal*, vol. 28, pp. 656–715, 1949.
13. B. Schneier, "Applied cryptography," *Second Edition, John Wiley & Sons*, 1996.

A Pipeline Architecture for Encompression† (Encryption + Compression) Technology

Chandrama Shaw[1] Debashis Chatterji[1] Pradipta Maji[1]
Subhayan Sen[1] B. N. Roy[2] P. Pal Chaudhuri[1]
[1]Department of Computer Science & Technology,
[2]Department of Electronics & Telecommunication
Bengal Engineering College (D U), Howrah, India 711103,
{cshaw,pradipta,subhayan,ppc}@cs.becs.ac.in

Abstract— **This paper introduces a new technology that combines compression and encryption into a single operation referred to as Encompression. This novel scheme is based on a special class of a sparse network - known as Cellular Automata (CA). The simple, regular, modular, cascadable, local neighborhood structure of CA ideally suits for low cost, online Encompression to support efficient data transmission with desired level of security. A pipeline architecture has been proposed for hardwired realization of Encompression operations.**

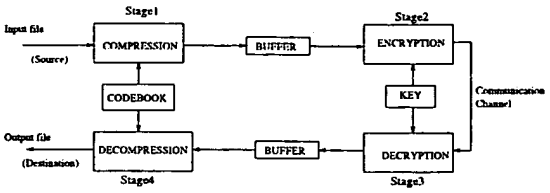


Fig. 1. Basic architecture of encompression

I. Introduction

The internetworked society of cyber age has been experiencing an explosion of data communication. The voluminous data of different type (text, image, video, audio, personal/business data) are transmitted over wired/wireless channels around the globe. The user community availing this facility are concerned about the hackers intruding into their data files transmitted over the public network. Further, notwithstanding the growth of channel bandwidth, it will always fall short of the demand for higher speed of transmission of larger volume of data from enlarged user community.

In the above background, efficient and secured data transmission demands due attention on following areas: (i) an efficient *compression* technique to reduce the data rate at source; (ii) an *encryption* scheme to ensure secured transmission; and (iii) low cost high speed execution of both operations.

Traditionally, the development of these two areas- compression and encryption have been undertaken by different research groups employing totally different techniques. However, a single technology supporting both *compression* as well as *encryption* is highly desirable for this internet age. *This paper bridges this gap and proposes a single technology - referred to as Encompression, supporting both encryption and compression.* The sole objective of this new technology is to provide support for efficient transmission of data with desired level of security.

.II. Encompression - An overview

Encompression covers two operations - *encryption* and *compression*. *Fig.1* shows the basic architecture of *Encompression* operations where the input data passes through the pipeline in a streaming mode. Currently, we have developed prototype *encompression* package to handle image and video files for which lossy compression techniques can be employed.

A static image or a video file is applied as an input to *Stage 1* (*Fig.1*). The data file is encoded through the *lossy compression* technique in this stage. The encoded data enter in a streaming mode to the next stage and gets encrypted by an *encryption* process in *Stage 2*. The *encompressed* data is transmitted through the communication channel. At receiver end, the *encompressed* data is first decrypted by the same key and original encoded data is retrieved in *Stage 3*. This encoded data is finally decoded in *Stage 4* by the decompression process. The *Stage 3* and *Stage 4* again operate with data streaming through these stages. Next two subsections present an overview of *compression* and *encryption* techniques employed for *Encompression* operation for video telephony/conference application.

A. Compression

Lossy Data Compression is a process of reducing the amount of data required to represent a given quantity of information with acceptable loss. It removes redundancy, repeatability and irrelevancy of data blocks of input file to generate compressed output. In order to demonstrate the capability of encompression technology, we have concentrated on *Lossy Compression* for *Video Telephony/Conference* application involving image of human portrait. This compression technique is next extended to compress a specific class of video files.

The well known *Vector Quantization* (VQ) [3] method has been applied to generate the *codebook* from the training set of human portrait. Even though considerable research has been done on the application of VQ for image compression, the scheme has not become popular for real life applications. The reasons for this non-acceptance lie on the following two inherent drawbacks of the scheme -

†The Patent Application No. 384/cal/2000 has been filed for En-

Proceedings of the 16th International Conference on VLSI Design (VLSI'03)
1063-9667/03 $17.00 © 2003 IEEE

good quality of image/video with high compression ratio; and (ii) the high processing overhead at encoding stage.

These two problems are addressed in this paper with the methodologies elaborated next. (i) Rather than developing general compression scheme for any arbitrary class of data files, we concentrate on developing the compression scheme for a specific class to improve the quality and *compression ratio*. We extract the domain knowledge from the specific class of data files - in the present case it is human portrait. This knowledge is next used to generate the *codebook*. *The same codebook will serve for other image/video data files which has the same characteristic of variation of pixel values.* (ii) The encoding time is reduced substantially by employing *Cellular Automata* (CA) technology as an *implicit memory*.

B. Encryption

Encryption is an effective way to protect data against eavesdropping. Again, the CA technology has been employed to design a *low cost, high speed Cryptosystem*. It employs a series of four-reversible CA transforms (*Fig.11*). on the *plain text* to arrive at the *cipher text*. The details of the scheme is reported in [5]. Only a summary of CA based encryption scheme is reported in *Section V* for space constrain. A brief introduction to CA theory follows.

III. Cellular Automata (CA)

A *Cellular Automaton* (CA) can be viewed as an autonomous finite state machine (FSM) consisting of a number of cells. The next state of a cell depends on its own state and the states of its right and left neighbors.

Linear/Additive CA: An n-cell Additive CA is characterized by an $n \times n$ characteristic matrix (T matrix) and an n-dimensional inversion vector F [1]. The elements of characteristic matrix T is represented as

$$T_{ij} = \begin{cases} 1, & \text{if next state of } i^{th} \text{ cell depends on } j^{th} \text{ cell} \\ 0, & \text{otherwise.} \end{cases}$$

and the inverse vector F is defined as

$$F_i = \begin{cases} 1, & \text{if next state of } i^{th} \text{ cell results from inversion} \\ 0, & \text{otherwise.} \end{cases}$$

The state transition behavior of an Additive CA can be characterized by the following relation :

$$f_{t+1}(x) = T \times f_t(x) + F(x) \qquad (1)$$

where $f_t(x)$ and $f_{t+1}(x)$ represent the CA states at t^{th} and $(t+1)^{th}$ instant of time respectively.

If all the states in state transition diagram of a CA lie in some cycles, it is a group CA; otherwise it is a non-group CA. Both group and non-group CA are employed in *encompression* technology.

Group CA can further be classified into *maximum* and *non-maximum* length CA. *Fig.2* shows the state transition diagram of a maximum length group CA; while *Fig.3* depicts a non-maximum length group CA.

The state transition graph of a non-group CA consists of a number of *cyclic* and *non-cyclic* states (*Fig.4*). The set of non-cyclic states form an inverted tree rooted at cyclic state (attractor). A CA whose state transition diagram

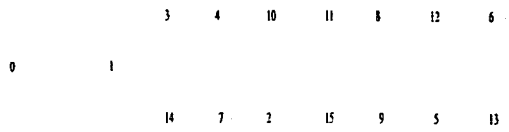


Fig. 2. A maximum length group CA

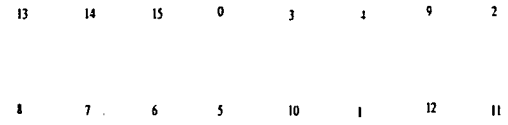


Fig. 3. A non-maximum length group CA

as *Multiple Attractor CA (MACA)* (*Fig.4*). Detailed characterization of $MACA$ is available in [1].

Non Linear CA: A CA employing non-linear rule with AND/OR logic, is known as non-linear CA. A non-linear group CA has been used in CA based cryptosystem (CAC) to realize non-affine transform.

IV. CA Based Lossy Compression

Vector quantization (VQ) is a lossy data compression method [3]. It maps the n dimensional vectors in the vector space R^n into a finite set of vectors, called the *codebook*. Each vector of the codebook is known as *codevector* or *codeword*. A cluster is the set of vectors having minimum deviation from the codevector. Thus each *codevector* is the nearest neighbor of the set of vector in a cluster. A VQ method mainly consists of two operations- (i) an *encoder* - to encode each block of input file with the index of a *codevector* in the *codebook*, and (ii) a *decoder* - to get back the representative block from the *codebook*.

The *encoder*, as shown in *Fig.5*, takes an input vector and outputs the index of corresponding *codevector* from the *codebook* that gives minimum deviation. The index of the *codevector* is sent to receiver end. The *decoder*, on receiving this index file, replaces each entry with the associated *codevector* found from the *codebook* kept on receiver side. *Codebook Generation* plays a key role in VQ scheme.

A. Codebook Design

Codebook design consists of two steps- (i) design of *training set*; and (ii) generation of *codebook*. Each step is illustrated with reference to *Fig.6*.

Design of training set: The training set has been designed out of 20 different human-face images with wide variation of pixel values. Each image of training set is segmented into 16 x 16 blocks that is subsequently processed in following three sequential steps:

Step1: Calculate *norm* (Standard Deviation) of each 16× 16 pixel block. If the *norms* match the pre-specified criteria (say $SD1$ as shown in *Fig.6*), it is stored as a member of 16 × 16 training set, else referred to as 16 × 16 residual

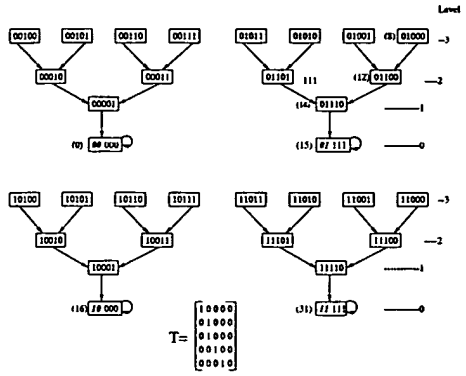


Fig. 4. A 5-cell non-group CA (MACA)

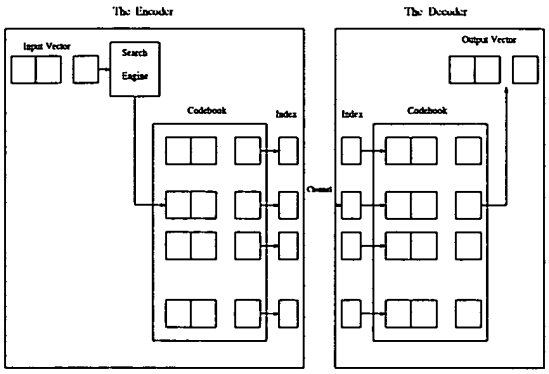


Fig. 5. Encoder and Decoder

Step2: Each member of residual 16 × 16 training set is broken into four 8 × 8 pixel blocks. Next, we calculate the *norms* of each 8 × 8 blocks and compare with the pre-specified criteria (say $SD2$ as shown in *Fig.6*). If the *norms* match the criteria, then it is stored as 8 × 8 training set. Otherwise it is referred to as the residual of 8 × 8 training set to be processed in the next step.

Step3: Each 8 × 8 residual block is broken into four 4 × 4 pixel blocks. Calculate the *norms* of each 4 × 4 blocks and compare with the pre-specified criteria (say $SD3$ as shown in *Fig.6*). If the *norms* match the criteria, then it is stored as 4 × 4 training set. Otherwise the blocks are discarded.

The matching criteria SD_i *(Standard Deviation)* has been fixed on the basis of statistical characteristics of 16 × 16, 8 × 8 and 4 × 4 pixel blocks of the training set.

Codebook Generation: To design the *codebook* from three training set (16 × 16, 8 × 8 and 4 × 4), we have used *Prune Tree Structured Vector Quantization (binary tree)* (*PTSVQ*) [3] method. Three *codebooks* are generated from three different training sets as shown in *Fig.6*. The *mean* value is computed for each training set. The *PTSVQ* is applied on mean removed vectors. Each element of the vector after subtraction of the mean value is known as *mean removed* vector. At the time of encoding, a 16 × 16 pixel block is taken from the image sequentially and depending upon the match criteria it is coded either by the *codebook* indices for 16 × 16 or broken to four 8 × 8 a 8 × 8 pixel block, if a proper match in the codebook is absent, it is treated as a collection of four 4 × 4 blocks and coded by four indices from the 4 × 4 *codebook*. A separate *match file* is kept to track the sequence of indices from different *codebooks*.

The VQ scheme we have implemented is a combination of mean removed VQ, $PTSVQ$, Classified VQ, and Address VQ (Partially). In binary $PTSVQ$, if N is total number of *codebook* entries, the depth of the tree is log_2N. So, time required to search each block is log_2N. As a result, if the *codebook* entry increases, the time required to search the best match increases. A significant improvement is needed to find the best match of a given input block with *codebook* entry. This is specifically true for on-line transmission of image and video data. A scheme based on CA technology is next reported to reduce the search time.

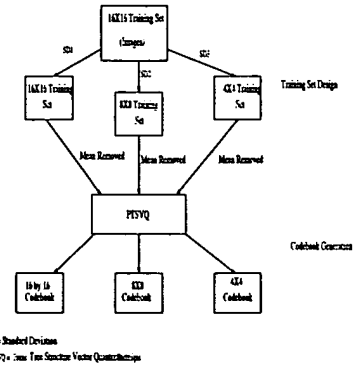


Fig. 6. Block diagram of codebook generation scheme

CA as Implicit Memory to Store and Search Codebook: To reduce the encoding time of VQ, we employ $MACA$ which effectively acts as a *codebook*. The details of $MACA$ characterization is available in [1]. $MACA$ is used to classify the set of patterns (codebook entries). The binary search for best match in codebook is implemented with an $MACA$ based multi-class classifier realized with multi-stage two class classifiers [6].

In order to identify the best match in binary $PTSVQ$ scheme, the input vector is compared with two centroid of two vector clusters in each layer of the tree and one of the branches is selected according to matching criterion. A vector cluster represents a set of entries in a *codebook*. A sequence of comparisons are done in subsequent levels till the leaf node is reached. We have designed $MACA$ based two class classifier to model the comparison operation at each node of $PTSVQ$ binary tree. The pixel blocks of the training sets employed for design of *codebook* and $PTSVQ$ binary tree is also used as input for design of $MACA$ based two class classifiers [6]. A set of $MACA$ are generated that acts as multi-class classifier of the vectors in a *codebook*.

Fig.7 illustrates the design of $MACA$ set for a *codebook*. Suppose, we want to classify the pattern set $C = \{\{S_0\},\{S_1\},\{S_2\},\{S_3\}\}$ into four classes - 0, 1, 2 and 3 such that the classifier would output correct class i (i = 0,1,2,3) for a given input *codevector* $P_i \in \{S_i\}$. At the first

and C_1, where $C_0 = \{\{S_0\}, \{S_1\}\}$ and $C_1 = \{\{S_2\}, \{S_3\}\}$. The $MACA$ (T_0) is designed to classify two distinct classes C_0 and C_1. *Fig.7* represents two classes C_0 and C_1 along with the $MACA$ (T_0). The same process is then applied for C_0 and C_1 to isolate $\{S_0\}$, $\{S_1\}$ and $\{S_2\}$, $\{S_3\}$ respectively and to generate two $MACA$s - T_1 and T_2. Thus the logical structure of multi-class classifier (*Fig.7*) is equivalent to $PTSVQ$ binary tree representing a *codebook*.

For a given codevector P_1 ($P_1 \in S_1$), we need to identify the *codebook* entry (that is the *codeword*) closest to P_1. At the prediction phase, the *codeword* P_1 is given as input and its class is identified as follows. At the first stage, the classifier designed with the $MACA$ (T_0) is loaded with P_1 and allowed to run. It returns the desired class C_0. In next level, the classifier of (T_1) is loaded with P_1 to output the class S_1.

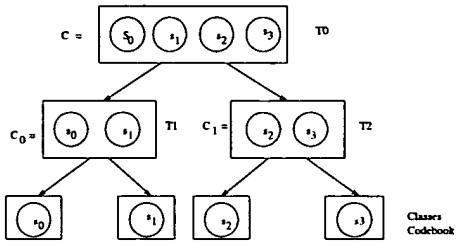


Fig. 7. Structure of multi-class clasifier equivalent to PTSVQ

Experimental Results of static images: The algorithm is applied on different standard pictures of human-face. Experimental results, reported in *Table I*, represent the $PSNR$ values as well as compression ratio of the set of images when compressed and decompressed using the proposed scheme. *Fig.8* shows the comparative study of original and decompressed images. The experimental results of *Fig.8* and *Table I* confirm high $PSNR$ value with a compression ratio in range of 98.50% to 98.83%.

TABLE I
RESULTS OF STATIC IMAGE

Image File	PSNR	Compression Ratio (%)
lena	33.36	98.83
girl	34.91	98.73
Proj10	32.69	98.64
Proj12	27.71	98.66
Proj17	29.26	98.50
Proj18	31.40	98.83
Proj19	29.04	98.70
Proj20	28.94	98.48
Proj21	28.68	98.25
Proj22	30.42	98.75

B. Application of Human-face codebook

On successful implementation of compression of still image of human face, we have explored the possibility of using the codebook generated for human-face in other application domains. We analyze the probability of occurance of pixel values of the training set of the human-face image. *Fig.9-*

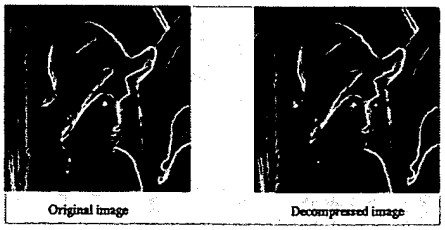


Fig. 8. Original and decompressed images

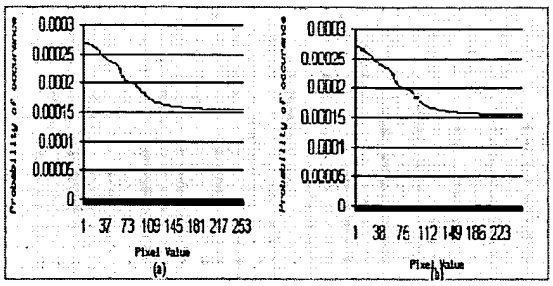


Fig. 9. Probability of occurrence of pixel values

value. This is computed by *Gaussian function* [7]. The other application domain which has characteristics similar to *Fig.9-(a)*, can use the same codebook for *compression.* We have experimented with two video files. From the 'toy video' file a set of frames are selected as training set. Probability of occurance of different pixel values, as noted in *Fig.9-(a)*, gets identified with *Fig.9-(b).* The close similarity of two groups confirms that the human-face codebook can be used in the application domain of 'toy video'. The experimental results of two 'toy video' is reported in *Table II* and *Fig.10.* The high compression ratio and good quality of video frames establish that the human-face *codebook* can be applied to other application domains having similar characteristics in respect of probability of occurance of pixel values.

V. CA BASED ENCRYPTION (CAC)

In general, the encryption employs a series of four-reversible CA based transforms (*Fig.11*) on plain text to arrive at cipher text. The basic idea of introducing a series of transforms - simple, moderately complex, and more complex - is to achieve desired level of security with minimum cost and high speed execution. Transform at each level is dependent on some function of the input key. The length of a key can be easily varied exploiting the modular and cascadable structure of CA.

The first level of encryption, (*Fig.11*) is a linear transform of the input key and also on plain text, the later one being implemented by rotating each bytes of the token. At the second level, an affine transform is introduced with an additive CA. A non-affine transformation is next implemented by employing a non-linear reversible CA [5]. Finally, a linear transform is employed with a simple XOR operation.

Acceptance of any cryptosystem depends on its crypt-

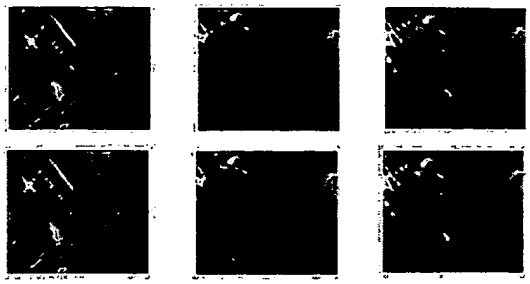


Fig. 10. Toy video sequence (original & decompressed)

TABLE II
RESULTS OF 'TOY VIDEO'

Video	No. of frames	PSNR	Compression Ratio (%)
toy1	100	31.14	98.66
toy2	50	32.44	98.50

differential cryptanalysis [2] and Shannon's notion of perfect secrecy test [4] on CAC. For the sake of comparison we also subject AES and DES to these two tests.

The experimental results of Differential cryptanalysis and Shannon's Security Quotient for CAC with the AES and DES are shown in *Table III* and *IV*. These results clearly establishe that the quality of security level of CAC is better than DES and comparable to that of AES. The CAC codesize is *3.20 KB*. It comes down below *3.0 KB* with hardware-software co-design [5]. The execution speed of CAC is noted in *TableV*. Thus the average throughput of the current version of encryption package is *2.20 Mbit/sec*.

VI. EXPERIMENTAL RESULT OF ENCOMPRESSION

The experimental result of each stage of *Encompression* operation is shown in *Fig.12* for the example image of *'lena'*. First, the image is compressed at *Stage*1 and the index file is generated. In *Stage*2, the index file is encrypted by the key '*abcdef*12345'. The decryption process decrypts the encrypted file by the same key and regenerates the index file (original) at *Stage*3. At final stage (*Stage*3), this file is decompressed and the image is retrieved. The size of the file at each stage is shown in *Fig.12* with its *ASCII character*.

VII. PIPELINED ARCHITECTURE OF ENCOMPRESSION

The pipeline architecture of *Encompression* hardware is shown in *Fig.14*. The *linear/additive CA* used for the *Encompression* can be realized out of the PCA structure of *Fig.13(a)*. Each such PCA has 4 switches and a MUX (Multiplexer) to configure any *linear/additive* rule on a CA cell. By contrast the *Universal Programmable CA* (UPCA) as shown in *Fig.13(b)* can be configured with *linear/additive/non-linear CA* rules. It requires 8 number of 2-to-1 MUX to incorporate the rule of a CA cell. The output of the MUX is the input to one 8-to-1 MUX.

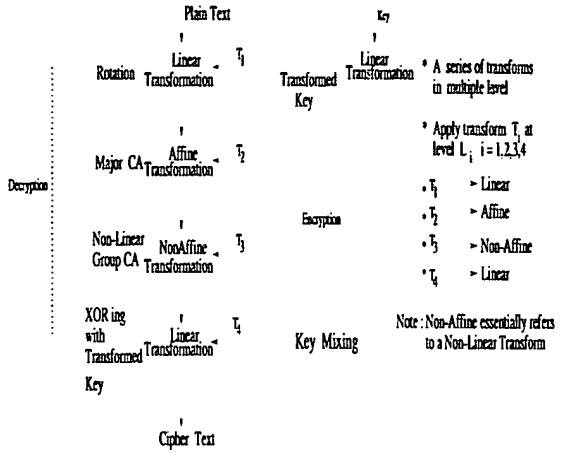


Fig. 11. Block diagram of encryption scheme

TABLE III
DIFFERENTIAL CRYPTANALYSIS

Input file Size (MB)	Avg. Std. Devi^n of XOR Distributions		
	CAC(%)	DES(%)	AES(%)
2	4.30	30.03	4.0
6	4.17	28.24	3.62
10	4.02	28.89	3.52
14	3.55	28.52	3.48
20	3.59	27.67	3.24

as the control input of this 8-to-1 MUX. While $UPCA$ is used for CAC scheme, the $MACA$ has been realized with PCA Different stages of *Encompression* hardware of *Fig.14* basically implements the architecture of *Fig.1* where data flows through the pipeline in streaming mode.

Stage1: This stage implements the $MACA$ based encoding sheme. Rule vectors of $MACA$ realizing the multiclass classifier are stored in *programme memory* as shown in *Fig.14*. The input block of image is stored in the input register. The control block configures $PCA1$ with rule vector and run for a cycle by taking the input register value. The output of $PCA1$ is stored in intermidiate register(inter_reg). The control block select next rule vector on the basis of the intermidiate register value. The process is continued until the class of the block is identified. The index of the identified input block is stored in buffer $B1$.

Stage2: The encryption block is enabled by *'enc'* signal when B1 buffer is filled with 128 bit value. The $PCA2$ is as an $UPCA$. It performs 4 level of operations as shown in *Fig.11* with the $B1$ value as token.Final encrypted token is generated and stored in buffer $B2$.

Stage3: The decryption process use the same circuit of encryption. The reverse process is implemented by activating the *'dec'* signal. The control block, configures $PCA3$ (an $UPCA$) to perform the 4 levels of operation in reverse order to retrieve original index. The retrieved index is stored in buffer $B3$.

Stage4: The index stored in $B3$ is used to address the *codebook* memory to read out the pixel block.

Proceedings of the 16th International Conference on VLSI Design (VLSI'03)
1063-9667/03 $17.00 © 2003 IEEE

TABLE IV

MEASUREMENT OF SHANNON'S SECURITY QUOTIENT

Input file Size (MB)	Shannon's Security Quotient ($\hat{\vartheta}$)		
	$CAC(\%)$	$DES(\%)$	$AES(\%)$
2	14.1605	14.2374	14.2345
4	10.1060	10.2507	10.1675
8	7.1182	7.1468	7.7046
13	5.5868	5.5645	6.0266
15	5.2097	5.3157	5.5552

TABLE V

COMPARISON OF EXECUTION TIME OF SOFTWARE VERSION

Input file (MB)	AES (sec)	CAC (sec)
2.78	2.52	1.20
7.68	8.67	4.21
11.80	14.10	6.90
14.81	18.10	8.80
23.76	29.53	14.61

simulated using *Cadence Verilog Simulator* on *Sun Ultra-60* machine. The design has been implemented with 0.25μ CMOS technology. The pre-layout area estimation of the non-optimized design is 4.25×10^6 sq. micron and timing analysis of one complete run of CAC is 1GHz. clock. The other modules (*Stage 1 & Stage 4*) are under implementation.

VIII. CONCLUSION

This paper presents a new technology known as *Encompression* that combines both *compression* and *encryption* into a single operation. The sparse network of CA has been employed for low cost, online *Encompression* to support efficient data transmission with desired level of security. A pipeline architecture of the proposed scheme is also outlined.

REFERENCES

[1] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chatterjee, "Additive cellular automata, theory and applications, vol. 1," *IEEE Computer Society Press, Los Alamitos, California*, no. ISBN-0-8186-7717-1, 1997.

[2] E. Biham and A. Shamir, *"Differential Cryptoanalysis of DES-like Cryptosystems"*, Journal of Cryptology, 4(1991), 3-72

[3] Allen Gresho and Robert M. Gray, *"Vector Quantization and Signal Compression"*, Kluwer Academy Publishers.

[4] C. E. Shannon, *"Communication theory of Secrecy Systems"*, Bell Systems Technical Journal, 28(1949), 656-715.

[5] Subhayan Sen, Sk. Iqbal Hossain, Kabirul Islam, Dipanwita Roy Chowdhuri, P Pal Chaudhuri, *"Cryptosystem Design for Embedded System Security"*, VLSI Design 2003.

[6] N. Ganguly, P. Maji, S. Dhar, B. K. Sikdar and P. Pal Chaudhuri, *" Evolving Cellular Automata as Pattern Classifier"*, Proceedings of Fifth International Conference on Cellular Automata for Research and Industry, ACRI 2002, Switzerland.

[7] M. R. Spiegel, J. Schiller and R. A. Srinivasan, *" Probability and Statistics"*, Schaum's Outline Series, McGraw-Hill International Editions, ISBN 0-07-118357-4.

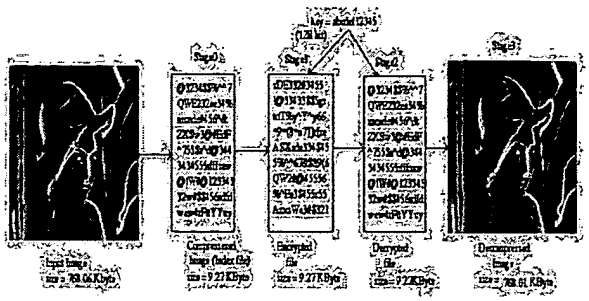


Fig. 12. Result of encompression at each stage

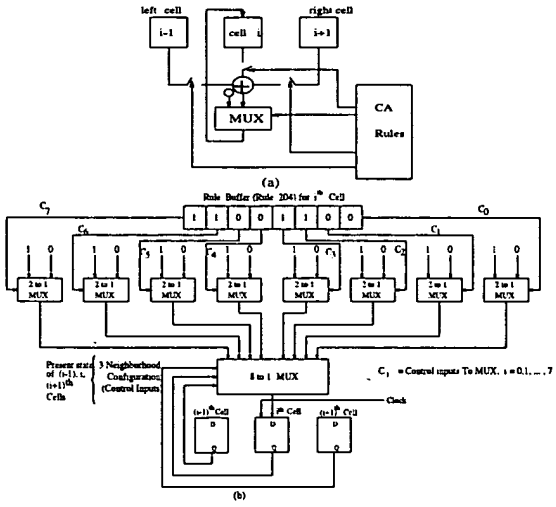


Fig. 13. Programmable CA (PCA) & UPCA cell structure

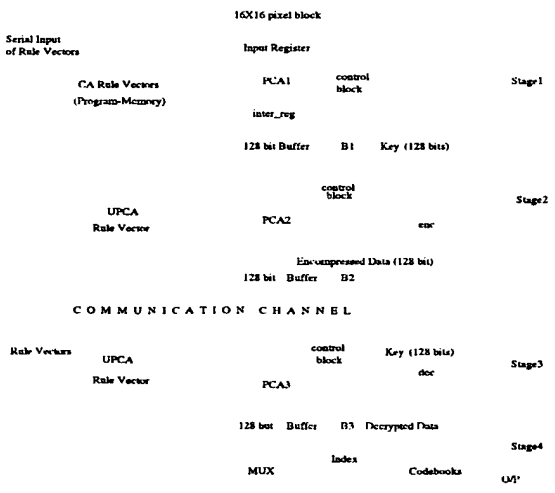


Fig. 14. Pipelined architecture of encompression hardware